Journal of
Applied
Crystallography

ISSN 0021-8898

Received 26 July 2010
Accepted 16 September 2010

AtomSim: web-deployed atomistic dynamics simulator

J. Brandon Keith,[a,b,c]* Jacob R. Fennick,[b] Daniel R. Nelson,[c] Chad E. Junkermeier,[b,c] Jiao Y. Y. Lin,[a] Chen W. Li,[a] Michael M. McKerns,[a] James P. Lewis[b] and Brent Fultz[a]

[a]Department of Materials Science, California Institute of Technology, Pasadena, CA 91125, USA, [b]Department of Physics, West Virginia University, Morgantown, WV 26506, USA, and [c]Department of Physics and Astronomy, Brigham Young University, Provo, UT 84602, USA. Correspondence e-mail: jbrkeith@gmail.com

AtomSim, a collection of interfaces for computational crystallography simulations, has been developed. It uses forcefield-based dynamics through physics engines such as the General Utility Lattice Program, and can be integrated into larger computational frameworks such as the Virtual Neutron Facility for processing its dynamics into scattering functions, dynamical functions *etc.* It is also available as a Google App Engine-hosted web-deployed interface. Examples of a quartz molecular dynamics run and a hafnium dioxide phonon calculation are presented.

1. Introduction

Atomistic simulations such as phonon calculations, free-energy optimizations, molecular dynamics (MD), Monte Carlo simulations and crystal structure prediction are powerful tools for interpreting experiments. However, as simulation codes in Fortran/C/C++ frequently are designed to be run from the command line and require inputs with detailed syntax, users often face a steep learning curve in installing, creating, staging and interpreting simulations. Also, since languages such as Fortran are geared toward the creation of monolithic executables, software architects must deal with human machine interaction issues while developing parsers and staging computations across a variety of architectures. So while the computational time and cost of materials simulations has decreased dramatically as a result of ever-advancing processor speeds and RAM size, the time and effort required to stage such simulations has remained approximately constant.

To lower this barrier we have implemented a user-friendly interface called *AtomSim* for atomistic simulation of crystallographic materials. It makes available in a single package a wide variety of forcefield-based techniques that are suitable for inorganic materials, organic materials and biomolecules. Its bindings include both Python and Java languages, each of which offers a different set of functionalities to users and software architects. For example, Python bindings provide the MD capabilities of the Molecular Mechanics ToolKit (MMTK; Hinsen, 2010), an open-source MD engine using primarily the Amber forcefield (Case *et al.*, 2005), and *GULP* (General Utility Lattice Program; Gale, 1997; Gale & Rohl, 2003), a general-purpose atomistic lattice simulation package with a large variety of customizable forcefields. The Java bindings interface not only the MD capabilities but also most of the functionality of *GULP*, including lattice dynamics (LD), free-energy optimizations, molecular dynamics, forcefield optimization and others described in §3. Additional physics and crystallographic engines can be added modularly.

AtomSim is built on modern object-oriented principles of software design to increase abstraction and componentization. This architecture has facilitated its deployment as an independent web application (Keith & Fultz, 2009) and as an integrated component in the Virtual Neutron Facility (VNF; Lin *et al.*, 2009), where users can embed MD/LD scattering kernels into samples that scatter neutrons and pipe simulation results to a variety of visualizers and analysis tools. The architecture and these two deployment environments are presented in §2. §3 describes the Java interface to *AtomSim*. We then

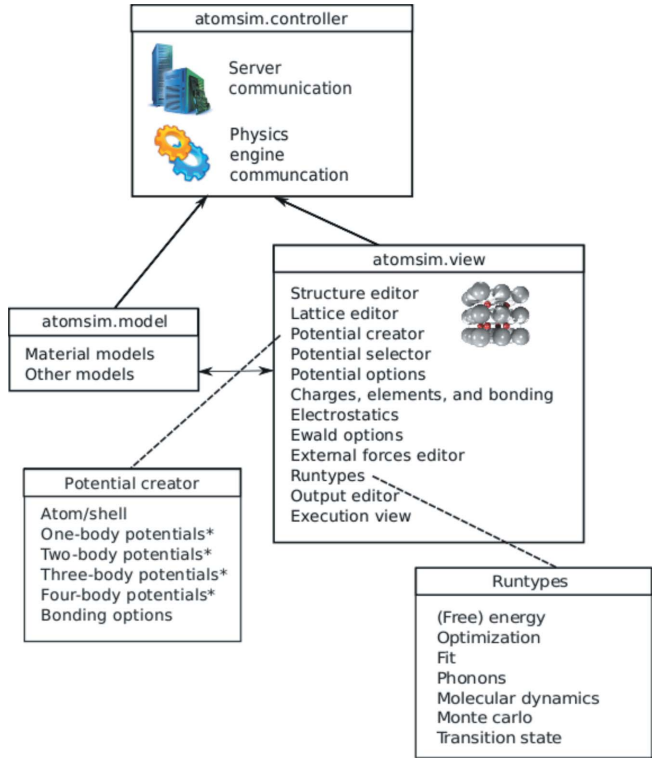

Figure 1
Schematic of *AtomSim* architectural organization. Items with an asterisk each contain a separate panel for the potentials listed in Table 1.

Table 1
Types of potentials found in *AtomSim*.

One-body potentials	
Spring	EAM functional
EAM density	Bond-order self-energy
Bsm	Cosh-spring
Two-body potentials	
General/Del Re	Buckingham
Lennard–Jones	Morse
Harmonic	Rydberg
Tsuneyuki	Squared harmonic
Tersoff	Tersoff combined
Stillinger–Weber two-body	Many body
EAM potential shift	Brenner
Spline	Qtaper
Qerfc	Bond-order charge
Polynomial	Swjb 2
Coulomb	Igauss
Covexp	Fermi–Dirac
Lennard–Jones buffered	Qoverr2
Damped dispersion	
Three-body potentials	
Three-body (harmonic)	Three (exponential harmonic)
Vessal	Cosine harmonic
Urey–Bradley	Murrell–Mottram
Linear three-body	Axilrod–Teller
Exponential	Stillinger–Weber three-body
Stillinger–Weber three-body (JB)	Bond–bond-cosine(angle) cross term
Bond–bond cross term	Bond–bond-angle cross term
Hydrogen bond	Equatorial
Four-body potentials	
Torsional	Out of plane
Ryckaert–Bellemanns	Torsion-angle cross potential
Harmonic torsional potential	Exponentially decaying torsional potential
Torsional potential w/tapering	ESFF torsional potential w/tapering
Inversion	

demonstrate *AtomSim* for simulations on quartz and hafnium dioxide in §4.

2. Architecture and deployment

Architectually, the source code of the *AtomSim* Java user interface (UI) is organized on the model–view–controller paradigm (Burbeck, 1992), where table models and other data objects are present in `atomsim.model`, Swing UI elements are present in `atomsim.view`, and `atomsim.controller` facilitates interaction between the UI and the underlying MD engine. This is shown in Fig. 1, which also displays various run types and potentials. Each of the *n*-body potentials listed in the potentials panel has another 10–20 potential panels associated with it, totaling the 58 potentials shown in Table 1. The grouping in the schematic roughly mirrors package organization in the source code.

As an example of the Python bindings, we show an inheritance diagram for some of the classes in the MMTK wrapper in Fig. 2. One can see that the main MMTK wrapper, `memd.mmtk.MMTK.MMTK`, inherits from `pyre.components.Component.Component`. This is a component in the Pyre runtime framework (Aivazis *et al.*, 2009), which facilitates the integration of diverse software components, manages component and application life cycles, and provides ease of extension of these components or applications to parallel and distributed computing (Cummings *et al.*, 2002). An important benefit of using this framework is that one may configure *AtomSim* from the command line and switch physics engines dynamically at runtime.

2.1. Stand-alone deployment

AtomSim can be accessed directly from the internet using several methods. The advantage of online deployment is that updates happen automatically each time the application is accessed, and the user is not required to install the application locally. The latter feature may be useful in corporate environments where users may not have permission to install software locally. These features are analogous to those found in the software-as-a-service paradigm. The Java bindings to *AtomSim* are deployed using Java Web Start, which caches the application to allow offline use. The Python bindings can be obtained through the instructions on the *AtomSim* home page.

2.2. VNF integration

AtomSim is also integrated into VNF. VNF provides an opportunity to interpret neutron scattering data using computational materials science in a user-friendly online environment (Fig. 3). It is a web-based architecture that stores data objects and computational results in a database for fast retrieval, searching and sharing with collaborators. VNF also connects to a variety of computational clusters where *AtomSim*-generated calculations can be run in parallel. It has visualizers and post-processors for viewing and analyzing many results from *AtomSim*.

Simulation of neutron scattering is done through bindings to *MCViNE* (Lin & Fultz, 2010), a Monte Carlo simulation of neutron

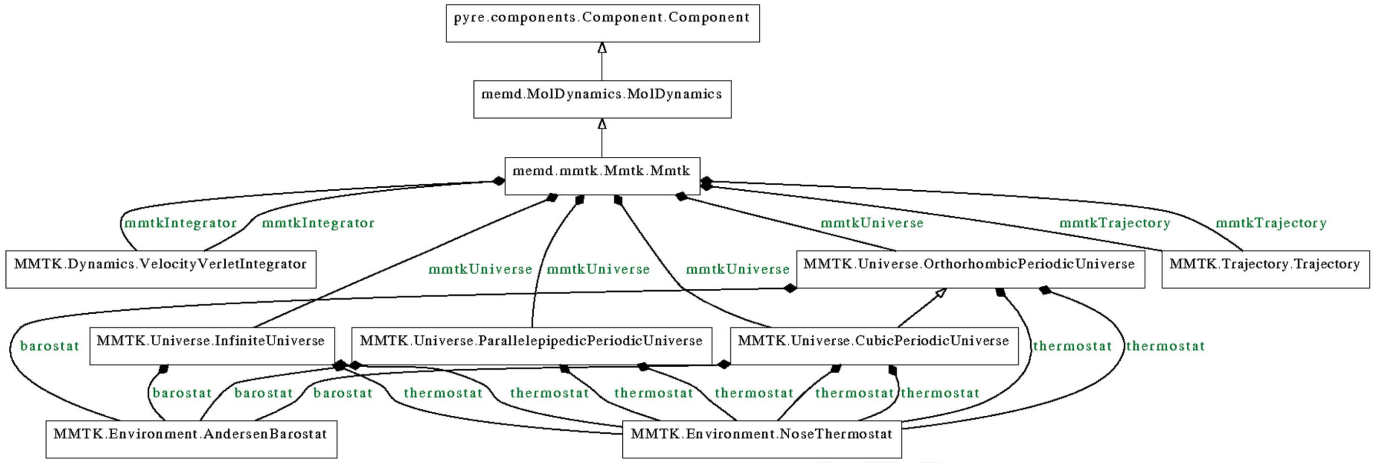

Figure 2
Inheritance diagram of major classes in MMTK bindings. The bindings are based on a Pyre component.

paths from a moderator through a sample to a detector. It allows a thorough treatment of all scattering physics at the sample and a complete, instrument-specific neutron spectrum. *AtomSim* is used in these simulations to configure forcefield-based physics engines to simulate the scattering probabilities as the neutron interacts with the sample.

In addition to integrating materials theory, instrument simulation and experimental data in the same computing environment, VNF provides programmatic access to data repositories such as the Crystallography Open Database (Chateigner *et al.*, 2007). Other capabilities include tools to build sample containers and virtual neutron instruments, and to visualize scattering results. The availability of these tools from within the same dynamic scripting environment allows users to design more efficient experiments by use of prior simulations, and to better understand the results of measurements through simulation, even while measurements are underway and experimental trends begin to emerge.

The integration of *AtomSim* in VNF is through Common Gateway Interface interaction. Results are retrieved from the server in JavaScript Object Notation format and converted to the internal data structures of *AtomSim*.

2.3. Availability and development

AtomSim is released under a BSD-style license (http://danse.us/trac/all/wiki/license) and is available for checkout under Subversion (http://dev.danse.us/trac/MolDyn/browser). Future versions of *AtomSim* will likely be a rich internet application using the Luban framework (Lin & Aivazis, 2010). Additional engines to be included in the future might be *LAMMPS* (Plimpton, 1995) and *GROMACS* (Lindahl *et al.*, 2001). During preparation of this document, *GULP 3.4* has been released with many new features. *AtomSim* is substantially compatible with *GULP 3.4* and will eventually be updated to include many of its new features.

3. Dynamically deployed UI

3.1. Algorithmic features

AtomSim makes available a unique set of capabilities for forcefield simulations. First, through the *GULP* engine, *AtomSim* offers the shell model for all simulation types. The shell model is a simplification of quantum polarization theories. It represents atoms with a positive point-charge core and a negative massless shell covered by charge. Shell and core are connected by a spring, giving atoms the ability to polarize when acted upon by external fields. Second, *AtomSim* provides controls for constrained optimizations, forcefield fittings and phonon dynamics with crystallographic symmetry. This leads to improved computational efficiency. Third, *AtomSim* offers in its potential panels 58 distinct interatomic potentials, listed in Table 1. These algorithmic capabilities are highlighted in the following description of the UI.

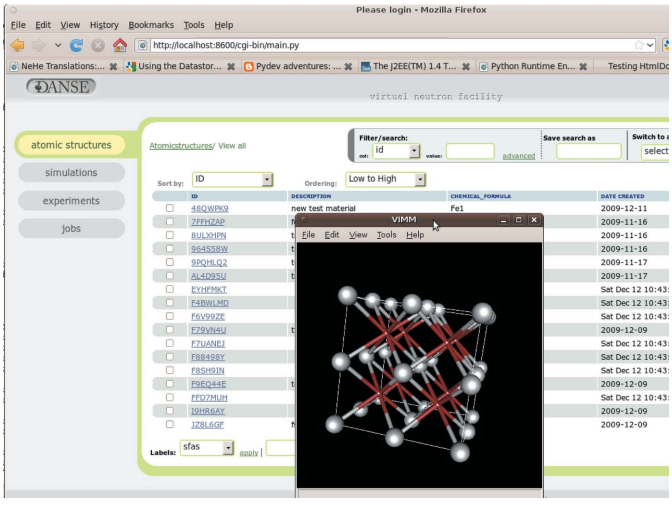

Figure 3
Screenshot of the beta version of the Virtual Neutron Facility (VNF).

3.2. General aspects of *AtomSim* UI design

AtomSim uses operating-system-independent Java with Swing components. The UI consists of several tabs, approximately arranged in the order a user would set up a calculation. Of particular interest is the `run type` tab, which is used to specify the type of calculation to be performed. The general schematic of the *AtomSim* architecture given in Fig. 1 outlines some of these panels. Within each panel are a series of options, each of which is given a border and title and contains text boxes, checkboxes, radio buttons and sliders. The user may enter information for *any* option *or* leave it blank. In general, all non-grayed text boxes of a given option must be completed, or *AtomSim* will prompt the user. Gray text boxes can be input optionally and have defaults when available. We now describe some of these panels.

3.3. Setting up and controlling the calculation

3.3.1. Atomic coordinates and crystal symmetry. The first tab is for entering atomic coordinates. Users can enter them manually or by importing from an xyz file. Inputs can be fractional coordinates, Cartesian coordinates or simply atomic unit-cell contents without specific positions as needed for crystal structure prediction. Users may also export the species and coordinates into an xyz file. Alternatively, if *AtomSim* is launched from within VNF, the atomic coordinates and unit-cell parameters will be transferred automatically to these panels from a user's database store in VNF.

In the `unit cell and symmetry` tab, users may specify the unit cell either as $a, b, c, \alpha, \beta, \gamma$ parameters or as Cartesian vectors. The crystal system and space group may also be set. The space group can be entered by number, by Herman–Mauguinn notation or by each symmetry operator. Options about turning symmetry on and off during different parts of the calculation may be specified. Supercells can also be specified, such as for MD runs or some phonon calculations.

3.3.2. Atomic and molecular potentials. *AtomSim* contains a number of libraries of potentials that ship natively with *GULP 3.1*, as well as user-supplied libraries. These are listed under the `potentials` tab under `use library`. A user only needs to click on a given potential to set it for a given calculation. The contents of each potential are shown to the right of the list of potentials. If *AtomSim* is launched from within VNF, potentials are stored in a database and retrieved at launch time. Users may upload new potentials with the `upload` button and share them with collaborators.

To create a new library, there is an experimental tab containing utilities for inputting potentials and saving them in libraries. One can input one-, two-, three- and four-body potentials as shown previously in Table 1. Potentials consist of choosing how many atoms are involved in the potential (1–4), which of the 58 types of potential to use, and which species and species type (core/shell) are involved. The main potential panel will then change accordingly. For conciseness, we only describe the pattern.

First, the mathematical form of the potential is displayed, followed by text boxes for each parameter including cutoff radii. Checkboxes

are also displayed, which allow the user to fit specific parameters. For two- and three-body potentials, bond connectivity diagrams are displayed to help users identify the parameters of each atom. Multiple potentials can be added by clicking on the `add potential` button. The user will be alerted to any errors.

General potential options may be entered from the `potential options` panel, including cutoffs of bond-length analyses, overriding interatomic potentials, dipole settings, scaling transformations, changing the Columbic forces within molecules and many others. A variety of output options, such as identification of valid three- and four-body terms, is also possible. Other panels exist for inputting charges and bonding options, electrostatics options, and Ewald summation options.

3.3.3. Energetics and external forces. Energetics options can be input under the `energetics` panel. Options are grouped into those related to energy and material properties and those related to the free energy. One of the strengths of the *GULP* engine is the capability for the user to substitute the free energy in various run types such as optimization by clicking the `calculate free energy instead of energy` textbox. If users choose to do this, they must specify a *k*-point mesh over the Brillouin zone and temperature. An additional panel is available for applying external forces to various atoms during the simulation, which is useful for directing dynamics.

3.4. Types of runs

Many different types of calculations useful to crystallographers are available in *AtomSim*, including structural optimization and fitting, phonon and free-energy calculations, molecular dynamics or Monte Carlo simulations, reaction barrier calculations, and crystal structure prediction. We describe a few of them below.

3.4.1. Optimization and fitting. Clicking on `run type` and inspecting the default `optimization` run type allows users to specify a number of options for their optimization. For example, users can switch between four different optimizers at specified points in the calculation. There are options on how to initialize, update, constrain and output the Hessian, as well as line minimization options. Options also exist for eliminating imaginary LD eigenvectors and for convergence of the optimization.

The user may apply a variety of constraints such as constant volume and constant pressure, and arbitrary atoms may be fixed by going to the `structures` tab and clicking on `fix` in the `fix positions` column. Options also exist to allow unit-cell parameters to change independently of internal structural parameters and to fix atomic cores independently of the atomic shells (and *vice versa*), among others.

To fit external data, one may click on the `fit` run type to make fits to *ab initio* energies, elastic constants, bulk moduli (Reuss, Voigt and Hill definitions), shear moduli (Reuss, Voigt and Hill definitions), static and high-frequency dielectric constants, static and high-frequency refractive indices, piezoelectric constants, monopole and Born effective charges, phonon frequencies, entropies, or heat capacities (constant volume). To assign which parts of the potential should be fitted, go to `create potential (experimental)` and select a potential. For most potential parameters, an accompanying fit checkbox can be toggled to fit experimental data. One may also specify (1) what type of optimizer is desired for fitting, (2) displacement or derivative fitting, (3) output frequency, (4) function, gradient and parameter tolerances, and (5) step size and iteration maxima. Options also exist for fitting shells while relaxing both their position and radius.

3.4.2. Phonon and (free) energy calculations. An example phonon calculation is presented in §4, but here we note that densities of phonon states, phonon dispersion curves and gamma-point corrections can all be applied when performing a phonon calculation. In addition, many options and file types can also be input or output.

Free-energy calculations allow a user to calculate or optimize the free energy instead of the energy. To use this feature, users should first input phonon settings and then add the free-energy run type afterward, specifying any additional settings on that panel. Two or more run types may be specified by clicking on the `run type` panel and clicking the `add` button after selecting additional run types. An example is specifying an optimization followed by a phonon run type, and then a free-energy run type. This will calculate an optimized structure that will be used automatically as the starting point for a free-energy calculation.

3.4.3. Molecular dynamics and Monte Carlo runs. *AtomSim* can stage MD in a variety of ensembles including NVE, NVT and NPT. It can use velocity verlet, leapfrog verlet and predictor–corrector as integrators. Various aspects of MD including equilibration, production and sample frequency may be controlled. The temperature may also be scaled to slowly heat or cool the system. Shells may be turned on or off during MD and various CPU-saving techniques employed, such as potential interpolation or vector tables. Monte Carlo capabilities include translating or rotating atoms and molecules, inserting and deleting atoms and molecules, adjusting the chemical potential, and many others.

3.4.4. Other run types. *AtomSim* also allows users to predict crystal structure and quantify transition state barriers in solid state reactions. These run types are not discussed here.

3.5. Execution and file management

Lastly, the `output` panel allows input and output files to be catalogued, altered and stored. Advanced users can directly alter the input files of *GULP*, save them with a new name and use these altered files for job submission. A time limit and description may be provided, and a variety of output formats or files are available. The `execute` tab allows the user to specify where to execute the simulation (localhost, remote host or VNF) and how to execute it (directly or with a scheduler). For localhost and remote host executions, a job manager is also provided, although remote host execution has been temporarily disabled. We also note that, depending on where *AtomSim* is run from (http://atomsimweb.appspot.com or VNF), some of the job submission locales will be disabled. For example, VNF submission is only enabled if a user runs *AtomSim* within VNF.

Beneath the locale panel is a large panel containing options that change with locale, such as specification of physics engine binaries and workspaces if run under localhost. An experimental feature is rapidly executing a series of calculations, called high-throughput execution.

When the user has finished specifying settings, he or she may click `submit` to run the job on a specified resource. Users may inspect the results by clicking `view` next to `stdout file` on the `output` tab. We now discuss the underlying architecture that facilitates these complex user interactions.

4. *AtomSim* examples

4.1. Quartz crystal MD

GULP is distributed with a series of example calculations. We demonstrate an MD simulation on quartz using a shell model (example 16). We start by entering the atomic structure in `atomic coordinates` because the atomic structure is necessary when configuring options in other panels. Accepting the default `coordi-`

nate type of 3d fractional, we set the number of atoms to 3 and click set. This gives three lines to specify atomic coordinates, atomic charge and many other atom-related parameters, as shown in Fig. 4. Now we enter space-group information into the unit cell and symmetry tab, as shown in Fig. 5.

We input the potentials in the order in which they are given in example 16. The first is a Buckingham interaction between Si cores and O shells. We select the potentials tab and the create library (experimental) sub tab. We select the interacting atoms as Si core and O shell in the first row of drop-down menus. The list of potentials is divided into four drop-down menus according to the number of interacting atoms. The Buckingham potential is a two-body interaction so it is within the drop-down menu labeled 2. We select the Buckingham entry. In the box below the select potential drop-down menu row, we see the equation for the Buckingham potential and text boxes for the potential parameters. We set A to 1283.907, ρ to 0.32052, C to 10.66158 and outer cutoff to 12.0. We accept the default of 0.0 for inner cutoff. The background of the inner cutoff text box is gray to signify a default setting, although the value can be changed, as for any other text box. Once the potential is entered, we click add potential and see the screenshot of Fig. 6. The second Buckingham potential is entered similarly.

To enter the spring potential, we choose the core–shell option in the self-interacting potentials drop-down list (called 1). This enables only the first drop-down menu in atoms. We choose O core, set k_2 equal to 74.92 and click the add potential button.

The final potential is the three body potential. This time the first three drop-down menus in the atoms box will become active. We set the first one to Si core, the second to O shell and the third to O shell. We set θ_0 to 109.47 and k is set to 2.09724. The potential cutoffs are r_{12} inner, outer cutoff = 0.0, 1.8; r_{13} inner, outer cutoff = 0.0, 1.8; r_{23} inner, outer cutoff = 0.0, 3.5. We then click the add potential button. To enter charges either we can input them under the charges column in the atomic coordinates panel, or we can click on the charges, elements and bonding tab and set O core to 4.0, O shell to −2.86902 and Si core to 0.86902.

Under the run type tab we select molecular dynamics from the list of run types and choose NPT ensemble in the thermodynamic ensembles box. We set the thermostat and barostat parameters ($q_{\rm nose}$ and $q_{\rm press}$) to 0.05. In the time lengths box we set timestep length to 0.0001 ps, with an equilibration time of 0.5 ps, production time of 0.20 ps and sample frequency of 0.5 ps. In the temperature box we set temperature (K) to 300. In the MD with shells with mass box we set the species drop-down list to O core and write 0.1 in the shell/core mass ratio text box. In output formats we click write xyz trajectory with a filename of example16.xyz.

Going to output we may select the view button next to the input files to see the raw output. Some snapshots of the trajectory are shown in Fig. 7.

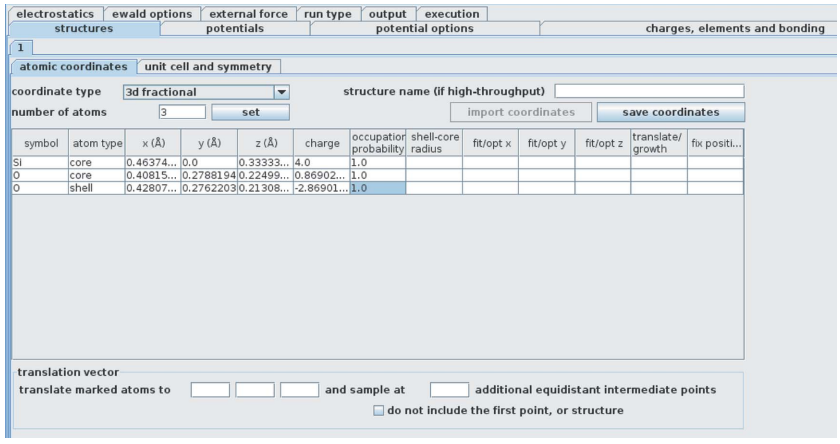

Figure 4
Atomic coordinates panel with SiO_2 input.

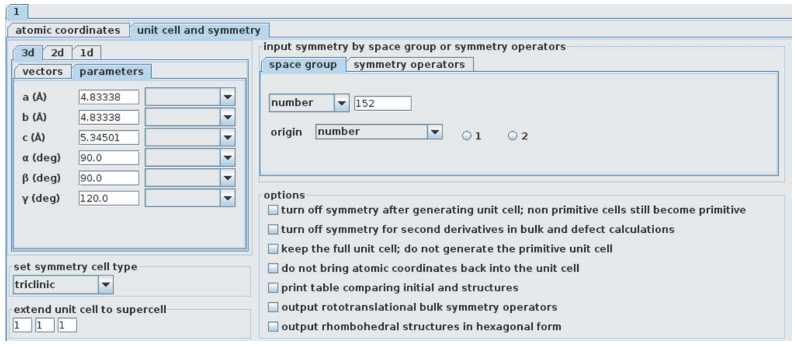

Figure 5
Space-group panel with SiO_2 unit cell.

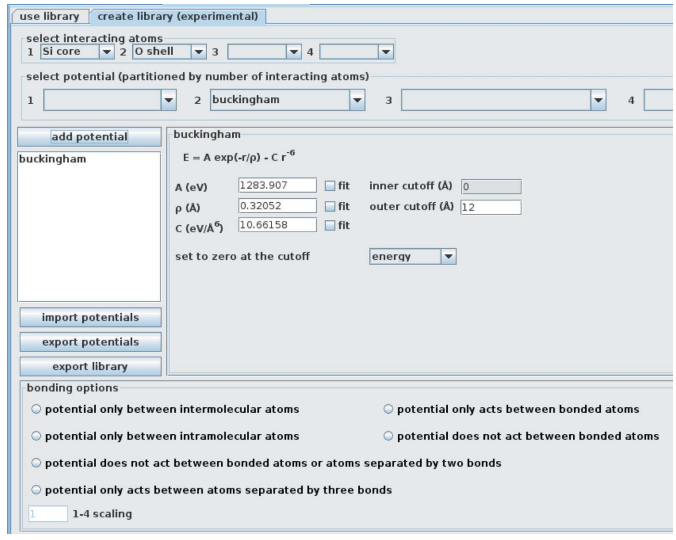

Figure 6
Inputing a Buckingham potential.

4.2. Hafnia phonon calculation

Monoclinic hafnia [hafnium(IV) oxide] is a widely used ceramic material with applications in coatings and high-κ dielectrics. We use hafnia as an example of how to perform phonon dispersion and density of states (DOS) calculations with a shell model (Li *et al.*, 2009).

We start with entering the information under atomic coordinates in the structures tab. Selecting 3d fractional under coordinate type, entering 5 in the number of atoms and pressing the set button

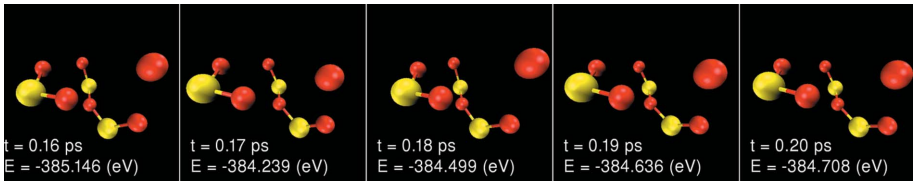

Figure 7
Snapshots of the quartz md trajectory.

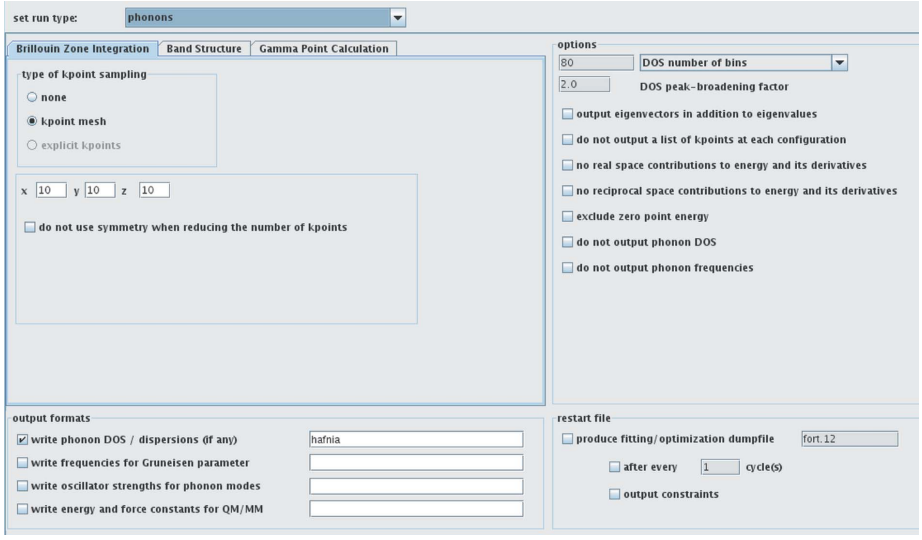

Figure 8
Inputing phonon settings for DOS calculation.

will enable the specifications table. We then input the five non-equivalent atoms and shells in the monoclinic structure of hafnia. To do so, we enter Hf, O, O, O, O in the `symbol` column, then select the `atom type` to be core, core, core, shell, shell and enter the atomic coordinates x, y, z in fractional units for each atom and shell, respectively. The result should be as follows: Hf core 0.276 0.040 0.208; O core 0.074 0.332 0.347; O core 0.449 0.758 0.48; O shell 0.0740 0.332 0.347; O shell 0.449 0.758 0.480.

Next we select `unit cell and symmetry`, `3d` and `parameters` to input the lattice constants as follows: $a = 5.117$, $b = 5.175$, $c = 5.291$, $\alpha = 90.00$, $\beta = 99.22$, $\gamma = 90.00$. To input the correct symmetry, we first set `set symmetry cell type` to `monoclinic`, then select `symbol` under the tab of `space group` and enter P21/C in the textbox on the right. To assign potentials between these atoms, we make use of the Lewis library (Lewis & Catlow, 1985). Under the `potentials` tab, we select `use library` and then choose `lewis`, which includes Buckingham potentials for hafnium and oxygen.

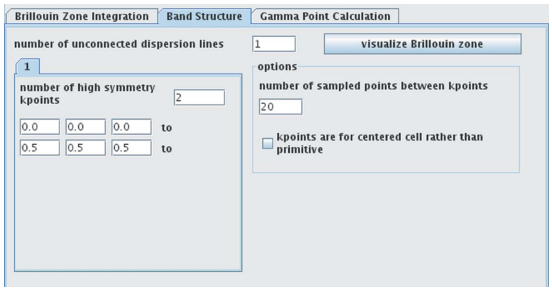

Figure 9
Inputing dispersion settings.

To perform a phonon calculation, under `run type` we set y as `phonons`. We note that cell geometry and atom coordinates typically need to be optimized before this step. For a DOS calculation, we choose `kpoint mesh` for `type of kpoints sampling` under the `Brillouin zone integration` tab and input $x = 10$, $y = 10$ and $z = 10$. The resolution of the DOS calculation can be specified by choosing `DOS number of bins`. We set this to 80, so the phonons panel should look like Fig. 8.

For a phonon dispersion calculation we click on the `band structure` tab, where one can create dispersion curves from an arbitrary number of k-point groupings. For now we leave the `number of unconnected dispersion lines` as one and the `number of high symmetry kpoints` as two. This will generate one dispersion curve between two k points. We input the first k point as 0.0 0.0 0.0 and the second as 0.5 0.5 0.5. The `number of sampled points between kpoints` could be altered, but we leave it at the default. We finish by selecting the check box in front of `write phonon DOS/disperisons (if any)` and add the filename `hafnia`. The dispersion panel should be similar to Fig. 9.

To check the generated input file, we go to the `output` tab and click `view` next to `gulp input files`. A pop-up window shows the *GULP* input file. Under `execution`, we choose `where to execute` the job and `how to execute` it. Clicking the `submit` button executes the calculation, which displays in the job monitor its status and when it is finished. Plotting the dispersion curves from the Γ point to the E point gives Fig. 10.

5. Conclusions

AtomSim is a convenient, web-accessible interface to atomistic simulations that has bindings to both Python and Java. Although it is primarily intended to support web services, it can be embedded in larger simulation frameworks (which themselves can be deployed as web services). *AtomSim* presently supports the packages MMTK and

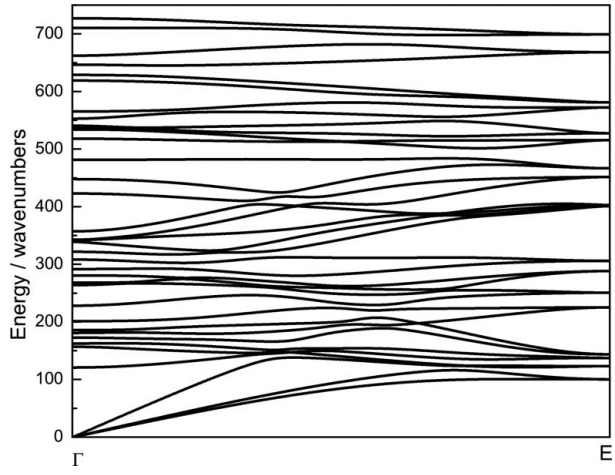

Figure 10
Phonon dispersions for hafnia from the Γ to E point.

GULP and can perform structural optimizations, lattice dynamics, free-energy optimizations, molecular dynamics and forcefield fitting. It can be extended to other packages such as *LAMMPS* and *GROMACS*. The complexity and richness of these software packages for MD simulations has required extensive development and testing of workflows that are natural for users, yet consistent with the input requirements of these packages. *AtomSim* is implemented with object-oriented programming techniques/principles and adheres to the model–view–controller architecture.

We thank J. Purewal and J. Desmarais for help in testing *AtomSim*. CEJ was supported by West Virginia Graduate Student Fellowships in Science, Technology, Engineering and Math (STEM), but is now a National Research Council Postdoctoral Associate at the Naval Research Laboratory. The initial concepts and pre-alpha release of *AtomSim* were supported by NSF grant CHE-036027. The alpha and beta release in VNF and the stand-alone web deployment release were supported under NSF grant DMR-0520547.

References

Aivazis, M., Lin, J. Y. Y., Keith, J. B. & McKerns, M. M. (2009). Pyre, http://dev.danse.us/trac/pyre.

Burbeck, S. (1992). *Applications Programming in smalltalk-80(tm): How to Use Model–View–Controller (mvc)*, http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

Case, D. A., Cheatham, T. E. III, Darden, T., Gohlke, H., Luo, R., Merz, K. M. Jr, Onufriev, A., Simmerling, C., Wang, B. & Woods, R. J. (2005). *J. Comput. Chem.* **26**, 1668–1688.

Chateigner, D., Chen, X., Ciriotti, M., Downs, R. T., Gražulis, S., Le Bail, A., Lutterotti, L., Matsushita, Y., Moeck, P., Olozábal, M. Q., Rajan, H. & Yokochi, A. F. T. (2007). Crystallography Open Database, http://www.crystallography.net.

Cummings, J., Aivazis, M., Samtaney, R., Radovitzky, R., Mauch, S. & Meiron, D. (2002). *J. Supercomput.* **23**, 39–50.

Gale, J. D. (1997). *JCS Faraday Trans.* **93**, 629–637.

Gale, J. D. & Rohl, A. L. (2003). *Mol. Simul.* **29**, 291–341.

Hinsen, K. (2010). MMTK, http://dirac.cnrs-orleans.Fr/MMTK/index.html.

Keith, J. B. & Fultz, B. (2009). *AtomSim*, http://atomsimweb.appspot.com.

Lewis, G. V. & Catlow, C. R. A. (1985). *J. Phys. C Solid State Phys.* **18**, 1149–1161.

Li, C. W., McKerns, M. M. & Fultz, B. (2009). *Phys. Rev. B*, **80**, 054304.

Lin, J. Y. Y. & Aivazis, M. (2010). *Luban: A Generic User Interface Language*, http://luban.danse.us.

Lin, J. Y. Y. & Fultz, B. (2010). *MCViNE: Monte Carlo Virtual Neutron Experiment*, http://danse.us/trac/MCViNE.

Lin, J. Y. Y., Keith, J. B., McKerns, M. M., Dementsov, A., Markovskiy, N., Aivazis, M. & Fultz, B. (2009). VNF, http://vnf.caltech.edu.

Lindahl, E., Hess, B. & van der Spoel, D. (2001). *J. Mol. Mod.* **7**, 306–317.

Plimpton, S. (1995). *J. Comput. Phys.* **117**, 1–19.